Une représentation en graphe pour l'enseignement de XML

Emmanuel Desmontils

Une représentation en graphe pour l'enseignement de XML

Emmanuel Desmontils

LINA, 2 rue de la Houssinière

BP92208, 44322 Nantes Cedex 03

emmanuel.desmontils@univ-nantes.fr

XML est un format actuellement très utilisé. Dans le cadre des formations en informatique, il est indispensable d'initier les étudiants à ce format et, surtout, à tout son éco-système. Nous avons donc mis au point un modèle permettant d'appuyer l'enseignement de XML. Ce modèle propose de représenter un schéma XML sous la forme d'un graphe mettant en valeur les caractéristiques structurelles des documents valides. Nous présentons dans ce rapport les différents éléments graphique du modèle et les améliorations qu'il apporte à la modélisation de données en XML.

Mots-clés: XML, Représentation graphique, Schéma, DTD, XSD, Relax NG.

Table des matières

Table des figures

Liste des tableaux

1. Motivation et objectifs

De nos jours, XML [8, 3] prend une place importante dans les systèmes informatiques. Ce format se retrouve aussi bien pour l'échange de données entre Web services, que pour le paramétrage d'applications ou pour mémoriser de façon pérenne des informations (à travers des bases de données XML [2] par exemple).

La complexité des documents est extrêmement variable. Même si beaucoup de structures XML (appelés schémas) sont simples, il est important de bien maîtriser la modélisation de tels documents. Il est souvent utile de s'appuyer sur des méthodologies de conceptions connues comme UML ou Merise [18, 4, 13, 10]. Cependant, ces méthodes ne sont pas vraiment satisfaisantes pour le cas des données hiérarchiques.

De plus, pour exploiter ou produire des documents XML, un développeur doit être capable de bien appréhender les schémas. Cela lui permet de tirer profit au mieux de la structure hiérarchique à travers les API dédiées ou les langages adaptés. Nous avons constaté par ailleurs que les utilisateurs de XML n'exploitent pas toujours bien la structure hiérarchique de ce format. Il est donc important d'introduire, dans la formation des développeurs, un enseignement sur XML et son éco-système, comme en particulier :

- les différents langages de schéma (DTD, XSD [16], Relax NG [5]),
- les API de programmation (SAX [14], DOM [21], etc.),
- les langages de requête et les bases de données XML (XPath [9, 6, 12], XQuery [9, 12], eXist [15], etc.),
- les langages de transformation (XSLT [20, 7]).

Durant les nombreuses années de formation à XML, nous avons constaté que, pour tous ces outils, le choix d'une représentation graphique permet de mieux appréhender la structure du document à exploiter ou à produire. Nous avons donc recherché une représentation sous forme de graphe pour mettre en évidence les caractéristiques structurelles du document à valider. Nous nous sommes inspirés des outils graphiques utilisés pour représenter les modèles relationnels.

Dans le suite, nous allons présenter des exemples utilisant uniquement le langage de schéma originel pour XML : DTD. Cependant, l'utilisation de XSD ou de Relax NG serait identique. En effet, nous attachons le plus d'importance à l'aspect structurel des schémas plutôt qu'au typage des informations.

Après avoir présenté notre exemple fil rouge (section 2) et les modèles existants (section 3), nous présenterons notre modélisation graphique pour les éléments (section 4), les attributs (section 5) et la structuration (section 6). Après une discussion sur ce modèle (section 7), nous conclurons.

2. Notre exemple

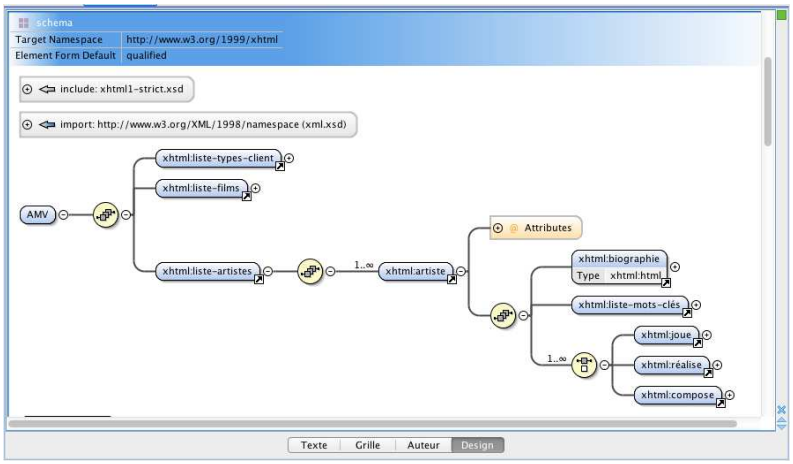
Afin d'illustrer notre modélisation, nous avons repris le sujet d'un exercice donné aux étudiants du Master MIAGE de Nantes en octobre 2013. Il concerne la modélisation d'un service (simplifié) de films à la demande auquel est adossé un réseau social spécialisé. Le texte décrivant le contexte, ainsi que la DTD associée sont donnés en annexe A et B. Le

modèle complet qui peut être produit à partir de ce schéma est donné en annexe C.

3. Modèles existants

Les outils de manipulation dédiés à XML proposent couramment une représentation graphique pour les schémas XSD ou Relax NG. Les figures 1a et 1b illustrent les représentations graphiques des schémas XSD [1] avec respectivement oXygen [2] et XMLSpy [3]. La représentation graphique de Relax NG (quand elle existe) est quasiment identique. Ces représentations ont en commun une représentation sous forme d'arbre (ou de forêt d'arbres) avec la possibilité de déployer ou non certaines branches. La structure en graphe n'est donc pas clairement visible et les liens de composition des éléments ne sont pas très lisibles.

a)



b)

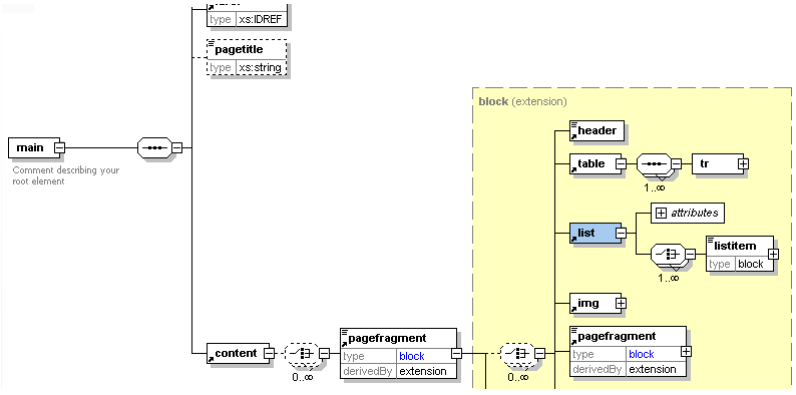


FIGURE 1: Visualisation XSD avec (a) oXygen et (b) XMLSpy

1. D'autres exemples de représentations graphiques de XSD peuvent être trouvées sur `http://en.wikipedia.org/wiki/XML_Schema_Editor`.
2. `http://www.oxygenxml.com/xml_editor/xml_schema_editor.html`
3. `http://www.altova.com/xmlspy/xml-schema-editor.html`

Pour les DTD, il n'existe pas de représentation graphique dédiée pour ce type de schéma.

Du point de vue général, il existe de nombreux outils graphiques de représentation des modèles conceptuels de données (diagrammes de classes pour UML [1], Entité-Association-Propriété pour la méthode Merise [19, 17], etc.). Ces modèles permettent de modéliser n'importe quelle structure de données, mais, de ce fait, ne sont pas nécessairement adaptés pour une compréhension des modèles hiérarchiques.

Cependant, parmi ces modèles standard, nous nous sommes intéressés aux "Crow's Foot diagrams" de G. C. Everest [11]. Ils sont utilisés par beaucoup de logiciels pour la représentation des tables et de leurs liens dans le modèle Entité-Relation[4]. La figure 2 présente un exemple d'utilisation de ce type de schéma pour la représentation d'un modèle relationnel. La forme graphique des cardinalités multiples (1..n ou 0..n) donne son nom à ce type de diagramme.

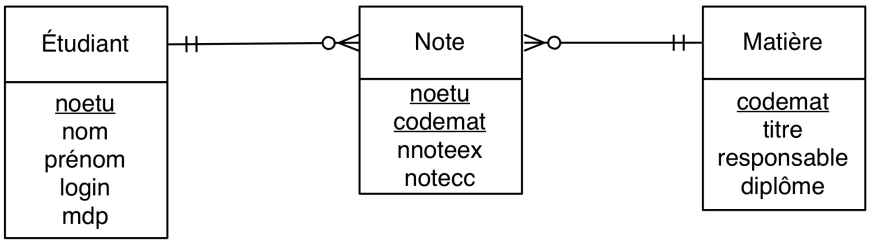


FIGURE 2: Exemple de "Crow's Foot Diagram" pour le modèle relationnel

Notre objectif est donc de proposer une représentation graphique quelque soit le schéma XML (DTD, XSD, Relax NG, etc.) utilisant ce type de notation et permettant d'avoir une représentation visuelle donnant une intuition de la structure hiérarchique des documents XML valides.

4. Modélisation des éléments

La notion d'élément XML est centrale, car elle amène le vocabulaire du dialecte et la grammaire. Dans cette première section de description du modèle, nous nous intéresserons uniquement à des éléments simples. Les éléments forment les sommets du graphe. Les liens entre les éléments correspondent aux contenus des éléments décrits par le schéma. Ces liens seront décrits en section 6.

4.1. Élément vide

Un élément est définit en DTD par `<!ELEMENT nom-élément contenu>`[5] où "contenu" est une expression s'apparentant aux expressions régulières et qui décrit la structure du

4. `http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model`

5. Ici, comme dans la suite de ce document, les extraits de schéma en DTD seront présentés avec la police `Courrier`.

contenu de l'élément. Dans notre modèle, un élément est représenté par un rectangle vert contenant le nom de l'élément. La figure 3 représente un élément vide décrit par `<!ELEMENT like EMPTY>`.

FIGURE 3: Élément vide

4.2. Élément à contenu textuel

Certains éléments n'ont pas de contenu structuré : leur contenu est le plus souvent textuel. Dans notre modèle, les textes seront représentés par un rectangle blanc. Éventuellement, ce rectangle contiendra un terme décrivant le type de texte contenu. La figure 4 représente un élément "like" avec un contenu textuel décrit en DTD par `<!ELEMENT like (#PCDATA)>`.

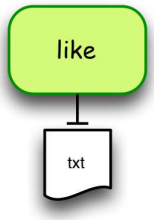


FIGURE 4: Élément à contenu textuel

Les éléments ne sont pas les seules sources d'information en XML : il y a aussi les attributs. Nous allons maintenant nous y intéresser.

5. Modélisation des attributs

5.1. Représentation d'un attribut

XML autorise le positionnement d'attributs associés aux éléments. La liste des attributs est représentée par un rectangle jaune. Les attributs ont un type (CDATA, ID, etc.) et un comportement (#REQUIRED, #IMPLIED, etc.). Le tableau 1 présente les cas d'attributs les plus fréquemment rencontrés et leur forme dans notre modèle.

TABLE 1: Formes d'attributs

	Forme	**Signification en DTD**
1	nom-cl	`nom CDATA #REQUIRED`
2	%date-modif	`date-modif CDATA #IMPLIED`
3	pseudo	`pseudo ID #REQUIRED`
4	#client	`client IDREF #REQUIRED`
5	#(clients)	`clients IDREFS #REQUIRED`
6	stars	`stars (0|1|2|3|4) #REQUIRED`
7	stars/'0'	`stars CDATA '0'`

Notons que, dans le cas d'une liste de valeurs possibles (ligne 6), notre modèle est incomplet. Ce n'est pas très grave au regard des objectifs de notre modélisation. Cependant, pour être plus complet, il est possible d'écrire "stars∈{0,1,2,3,4,5}". La figure 5 représente par exemple un élément vide décrit en DTD par `<!ELEMENT client EMPTY>` avec trois attributs :

- `<ATTLIST client pseudo ID #REQUIRED>`,
- `<ATTLIST client nom-cl CDATA #REQUIRED>`,
- `<ATTLIST client prénom-cl CDATA #REQUIRED>`.

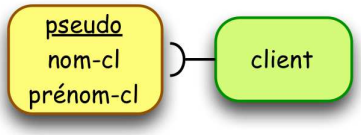


FIGURE 5: Attributs avec un identifiant

Les cas du tableau 1 peuvent être combinés. Par exemple, la figure 6 propose un attribut "{stars}/'0'" qui est un attribut pris dans une liste de valeurs (par exemple "0|1|2|3|4|5") et avec comme valeur par défaut "0".

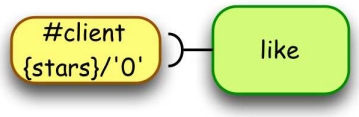


FIGURE 6: Attributs avec une référence

5.2. Liens entre attributs

Afin de préciser le rôle des attributs de type IDREF(S), il est possible d'ajouter une flèche en pointillé allant de l'attribut vers l'identifiant qu'il est supposé référencer. Les schémas XML ne prévoient pas de préciser ce lien, mais il facilite l'exploitation des documents DTD ou XSD (mise au point des API, utilisation des langages de recherche d'informations, etc.). La figure 7 montre le lien entre un attribut IDREF et l'attribut ID correspondant. Ici, l'attribut "client" de l'élément "like" doit contenir le "pseudo" d'un client.

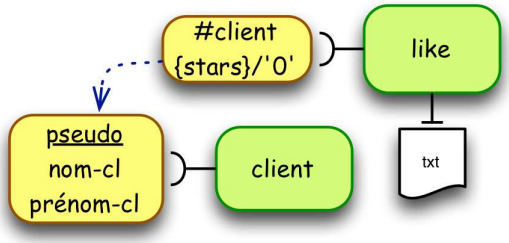


FIGURE 7: Lien entre ID et IDREF

La figure 8 représente le lien entre un attribut IDREFS (ici "acteurs") et l'attribut ID qui correspond ("pseudo").

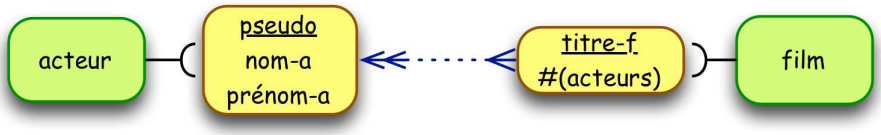


FIGURE 8: Lien entre ID et IDREFS

Ces liens ne font pas à proprement parler partie du graphe. Ils sont plus des commentaires, des appuis, pour les utilisateurs du graphe.

Maintenant que les noeuds du graphe ont été modélisés, nous allons nous intéresser à la modélisation des arcs.

6. Modélisation des contenus complexes

La structure de graphe n'apparaît que quand un élément en contient d'autres. Nous allons présenter les différents cas standard et élémentaires que nous pouvons rencontrer.

6.1. Itérations

Tout d'abord, nous allons représenter les opérateurs d'itération "*", "+" et " ?". Pour cela, nous nous sommes inspirés des "Crow's foot diagrams" de G. C. Everest [11] utilisés principalement pour représenter graphiquement les tables du modèle relationnel. La figure 9 présente la symbolique utilisée pour chacun des opérateurs. Ces opérateurs permettent de mettre en place les arcs élémentaires entre les différents sommets du graphe.

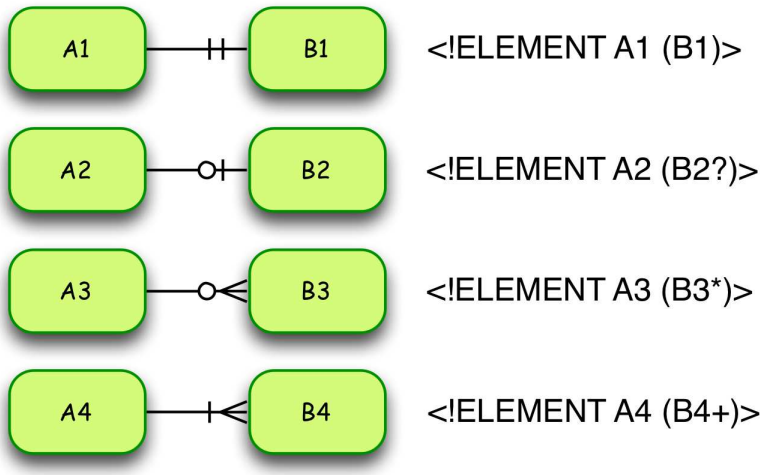


FIGURE 9: Itération d'éléments

6.2. Séquences et alternatives

Il reste deux opérateurs à introduire : la composition d'éléments en séquence et l'alternative. Chacun des éléments qui composent la séquence ou l'alternative font l'objet d'un des opérateurs d'itération présentés dans la section 6.1, d'une autre séquence, d'une autre alternative ou d'un sous-groupe présenté dans la section 6.3.

La séquence permet de décrire un contenu comme une succession d'éléments. Elle est représentée dans notre modèle par un point noir. L'ordre des nœuds est l'ordre dans le parcours trigonométrique autour de ce point en partant de la gauche du modèle. La figure 10 illustre une séquence et se traduit par `<!ELEMENT AMV (liste-clients, liste-films, liste-artistes)>`.

L'alternative permet de donner le choix entre plusieurs éléments. Elle est représentée dans notre modèle par une fourche. La figure 11 illustre une alternative et se traduit par `<!ELEMENT AMV (liste-client | liste-films | liste-artistes)>`.

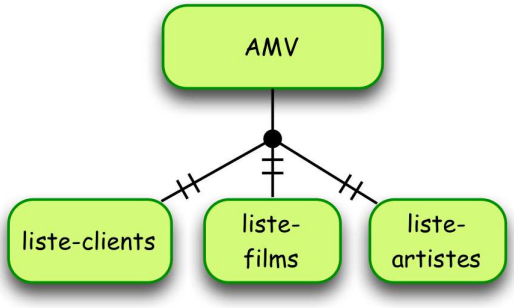


FIGURE 10: Composition d'éléments en séquence

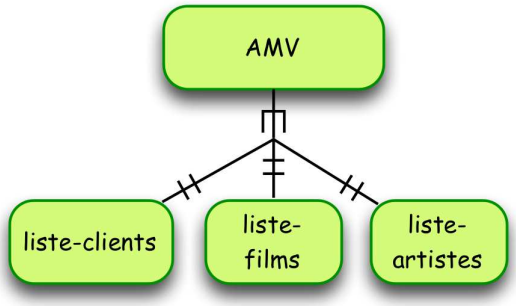


FIGURE 11: Alternative d'éléments

6.3. Sous-groupes

Certaines descriptions de contenu sont construites en utilisant des sous-groupes. Par exemple, supposons qu'un artiste puisse avoir été acteur dans certains films, metteur en scène dans d'autres, voir compositeur de bande originale. Alors, nous pourrions proposer la définition suivante : `<!ELEMENT artiste (joue | réalise | compose)+>`.

La partie `(joue | réalise | compose)` dans l'exemple ci-dessus est un sous-groupe sur lequel est appliqué l'opérateur d'itération "+" (vu dans la section 6.1). Les trois éléments sont alors répétés dans un ordre quelconque (alternative vus dans la section 6.2).

Dans notre modèle, un sous-groupe est mis en évidence par une zone orangée. La figure 12 illustre l'exemple ci-dessus. Cette notion de sous-groupe est "récursive" : un sous-groupe peut lui-même contenir des sous-groupes. Dans notre modèle, il y aura alors une imbrications de structures.

6.4. Schéma importé

Pour terminer, il est parfois utile d'importer le schéma pré-existant d'un dialecte spécifique ou d'un langage standard. Dans ce cas, il n'est pas toujours utile de le détailler, car secondaire ou, au contraire, parfaitement maîtrisé. Dans ce contexte, il n'est pas

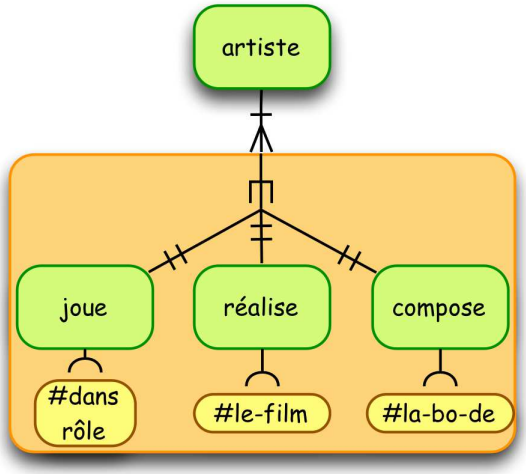


FIGURE 12: Sous-groupe d'éléments

nécessaire d'expliciter son graphe. Aussi, nous avons introduit dans notre modèle un symbole représentant un sous-graphe qui n'est pas détaillé. Pour cela, nous utilisons le symbole du nuage.

La figure 13 représente la biographie d'un artiste en XHTML[6]. Ce langage, bien connu, n'a pas besoin d'être représenté pour être manipulé. Il suffit alors de donner à "biographie" le même contenu que la balise "<body>" en XHTML (contenu qui est décrit par l'entité paramètre "%body;").

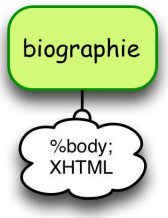


FIGURE 13: Contenu d'un élément identique à celui de <body> en XHTML

Cette simplification peut aussi permettre de ne présenter qu'un graphe partiel. La partie mise en ellipse peut être considérée comme sous-entendue ou comme un "sous-graphe". Le graphe principal est alors considéré comme un graphe hiérarchique, la partie sous-entendue peut faire l'objet d'une graphe secondaire. La figure 14 représente le contenu de l'élément "liste-films" comme une partie du graphe non développée.

6. La DTD de XHTML est décrite à l'URL : `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd`.

FIGURE 14: Contenu non-détaillé dans un élément

7. Discussion

7.1. Les faiblesses du modèle

Ce modèle a été mis au point à des fins pédagogiques. Il n'est donc pas adapté à des schémas très complexes que l'on peut trouver dans certains systèmes d'information. En effet, la topographie du graphe devient difficile à aborder sur une seule page A4 avec un grand nombre d'éléments ou lorsque le nombre des imbrications de groupes dépasse deux ou trois niveaux. Notre exemple complet en annexe C illustre bien cette idée. En effet, il ne comporte qu'une vingtaine d'éléments et semble, visuellement, déjà bien complexe. Par contre, il reste utilisable en conditions opérationnelles pour de petits schémas, souvent présents dans des applications Web à base de services.

Nous pouvons noter aussi que le graphe seul reste incomplet pour une vue exhaustive des éléments et des attributs. La description du contenu des attributs et des éléments reste très basique. Le modèle ne couvre même pas l'ensemble des types en DTD : les listes ne sont pas détaillées, les types NMTOKEN et NMTOKENS ne sont pas identifiés, tout comme le type NOTATION. Les entités générales ne sont pas non plus explicitées. Par contre, les entités paramètres le sont de manière indirecte (par leurs effets). Dans notre exemple en annexe C, le mécanisme d'inclusion de la DTD de XHTML s'effectue par ce principe.

7.2. Un modèle pédagogique

Malgré tout, ces schémas restent un appui de taille pour aborder la construction de chemins de localisation en XPath, pour comprendre le mécanisme de parcours par l'API SAX, les optimisations de parcours avec DOM ou l'application des règles en XSLT. En XPath, par exemple, il est plus facile de construire le chemin pour atteindre une information en se référençant à notre modèle. En particulier, les liens entre les attributs de type IDREF ou IDREFS et leur correspondant de type ID permettent de mieux comprendre les fonctions XPath "id()" (en suivant la flèche) et "idref()" (en remontant le flèche).

De plus, l'utilisation des symboles tirés des "Crow's foot diagrams" permet de donner une idée intuitive de la structure de l'arbre XML résultant. Pour renforcer l'intuition

de l'arbre résultant, il est conseillé de travailler aussi sur la topographie du graphe. En effet, dans la plupart des cas, le graphe est déjà un arbre, voir un treillis (comme dans notre exemple en annexe C), et il est utile d'organiser sa structure comme tel. Ainsi, l'organisation globale et les "crow's foot" donnent une idée de ce que sera l'arbre XML résultant.

7.3. Un nouveau pas dans la modélisation XML

Notre modélisation en graphe peut s'inscrire dans le processus de modélisation de données ou de connaissances sous forme hiérarchique, comme l'illustre la figure 15.

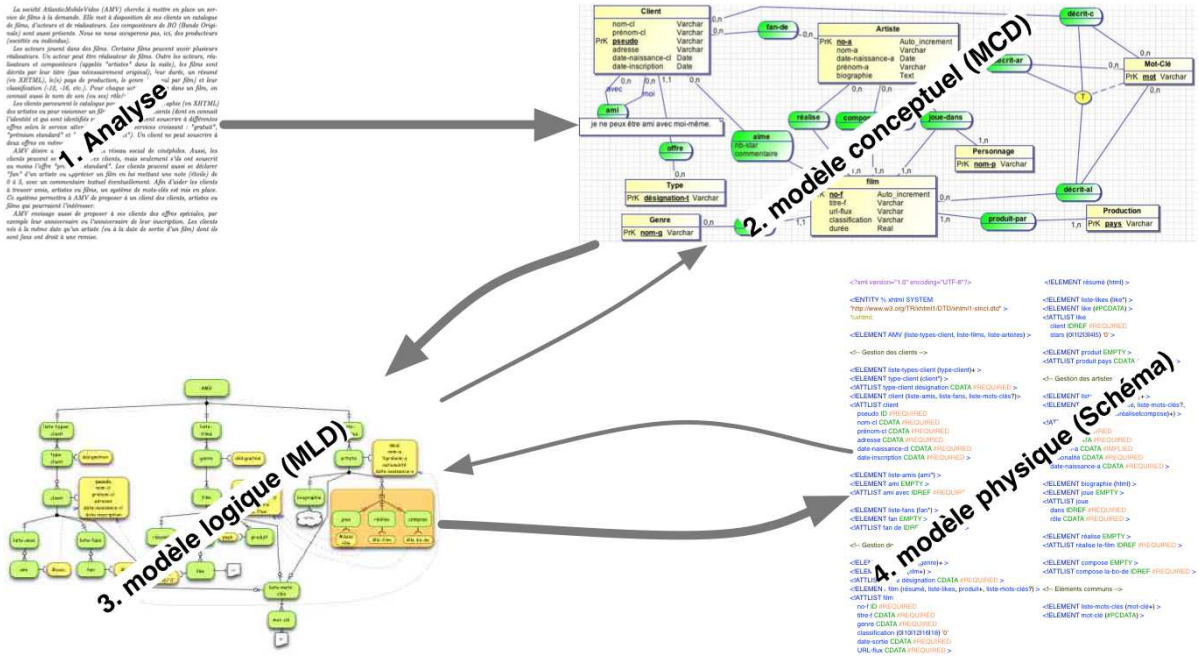


FIGURE 15: Processus de modélisation avec notre modèle

En effet, un processus de modélisation standard commence par l'analyse du problème. Cette analyse amène la construction d'un modèle conceptuel des données, en Merise ou en UML par exemple, puis à un modèle physique, dans notre contexte en XML [18, 4, 13, 10] (étapes 1, 2 et 4 dans la figure 15). Notre modèle peut s'insérer entre le modèle conceptuel et le schéma (étape 3 dans la figure 15). En effet, il peut être considéré comme le modèle logique des données, car il va mettre en valeur les propriétés du modèle physique XML (structure hiérarchique et contrôle syntaxique) et réaliser certains types de contraintes énoncées dans le modèle conceptuel.

Ainsi, comme dans notre exemple en annexe C, il est possible de profiter de listes de valeurs courtes (les types d'abonnement par exemple) pour les transformer en contraintes structurelles. De la même manière, les contraintes du modèle conceptuel peuvent parfois

être transformées en contraintes syntaxiques dans le modèle logique [10]. Dans notre exemple, la liste d'amis n'est possible que dans les abonnements "prémium".

Dans le cadre pédagogique uniquement, le processus de modélisation est alors pris à rebours en partant du schéma, objet de l'étude, pour remonter vers la modélisation en graphe pour mettre en évidence la structure.

8. Conclusion

Le modèle graphique de schémas XML que nous présentons ici est donc un modèle orienté vers l'aspect structurel du document XML et moins vers l'aspect type de données. C'est un modèle intéressant pour l'enseignement, surtout si le schéma n'est pas trop complexe. Il permet une bonne maîtrise du schéma, surtout pour la découverte de XPath, XSLT et des API de programmation (SAX et DOM). Il peut aussi être utilisé lors du processus de construction du schéma XML en jouant le rôle de modèle logique des données (entre le modèle conceptuel en Merise ou UML et le modèle physique qu'est XML).

Le modèle est opérationnel dans le cadre des enseignements de l'éco-système XML en Master à l'université de Nantes depuis quelques années. Malheureusement, actuellement, il n'existe pas encore d'outil logiciel permettant de gérer notre représentation, en particulier sa topographie, semi-automatiquement [7].

9. Remerciements

Nous tenons à remercier les étudiants de la MIAGE de Nantes (en présentiel et à distance) [8] ainsi que les étudiants de Master CCI de Nantes [9] d'avoir servi de cobaye pour ce travail pendant plusieurs années.

Références

[1] Grady Booch, James Rumbaugh, and Ivar Jacobson. The unified modeling language (UML). *World Wide Web : http://www.rational.com/uml/ (UML Resource Center)*, 94, 1998.

[2] Ronald Bourret. XML and databases, 1999. URL `http://ece.ut.ac.ir/dbrg/seminars/AdvancedDB/2006/Sanamrad-Hoseininasab/Refrences/6.pdf`.

[3] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (XML). *World Wide Web Journal*, 2(4) :27–66, 1997.

7. L'ensemble des figures de ce rapport ont été conçues à l'aide du logiciel de dessin vectoriel OmniGraffle (`http://www.omnigroup.com/omnigraffle`).

8. `http://miage.univ-nantes.fr/`

9. `http://www.dpt-info.univ-nantes.fr/44376489/0/fiche___pagelibre/`

[4] David A Carlson. *Modeling XML applications with UML : practical e-business applications*. Addison-Wesley Reading, 2001.

[5] James Clark and Makoto Murata. Relax ng. *Committee Specification*, 20011203, 2001. URL `http://relaxng.org/`.

[6] James Clark, Steve DeRose, et al. Xml path language (xpath), 1999.

[7] James Clark et al. Xsl transformations (xslt). *World Wide Web Consortium (W3C).*, 1999. URL `http://www.w3.org/TR/xslt`.

[8] World Wide Web Consortium et al. Extensible markup language (xml) 1.0. *W3C XML, February*, 1998. URL `http://www.w3.org/XML/`.

[9] World Wide Web Consortium et al. XML path language (XPath) version 1.0, 1999. URL `http://www.w3.org/TR/xpath/`.

[10] Emmanuel Desmontils. Modélisation avec XML, cours e-miage, 2005. URL `http://miage.univ-nantes.fr/miage/D2X1/chapitre_modelisationXML/chapitre.htm`. in French.

[11] G. C. Everest. Basic data structure models explained with a common example. In *Proc. Fifth Texas Conference on Computing Systems*, pages 18–19, 1976.

[12] Mary Fernández, Ashok Malhotra, Jonathan Marsh, Marton Nagy, and Norman Walsh. XQuery 1.0 and XPath 2.0 data model. *W3C working draft*, 15, 2002. URL `http://www.w3.org/TR/xquery/`.

[13] Antoine Lonjon and Jean-Jacques Thomasson. *Modélisation XML*. Editions Eyrolles, 2006. in French.

[14] David Megginson and David Brownell. Sax : The simple api for xml. *online material, available at* `http://www.megginson.com/SAX/index.html`, 2001. URL `http://www.saxproject.org/`.

[15] Wolfgang Meier. eXist : An open source native XML database. In *Web, Web-Services, and Database Systems*, pages 169–183. Springer, 2003. URL `http://exist-db.org`.

[16] XML Schema Part. 0 : Primer. *W3C Recommendation*, 2, 2001. URL `http://www.w3.org/XML/Schema`.

[17] Pham Thu Quang, Cyrille Chartier-Kastler, and David Davison. *MERISE in Practice*. Macmillan, 1991.

[18] Nicholas Routledge, Linda Bird, and Andrew Goodchild. UML and XML schema. *Australian Computer Science Communications*, 24(2) :157–166, 2002.

[19] Hubert Tardieu, Arnold Rochfeld, René Colletti, and Jacques Lesourne. *La méthode MERISE : principes et outils*. Editions d'organisation, 1983. in French.

[20] XSL Transformation. Version 1.0, w3c recommendation 16. november 1999, 1999. URL `http://www.w3.org/Style/XSL/`.

[21] Lauren Wood, Arnaud Le Hors, Vidur Apparao, Steve Byrne, Mike Champion, Scott Isaacs, Ian Jacobs, Gavin Nicol, Jonathan Robie, Robert Sutor, et al. Document object model (DOM) level 1 specification. *W3C Recommendation, October*, 1998. URL `http://www.w3.org/DOM/`.

Annexe A : Description de l'exemple

En octobre 2013, les étudiants de Master MIAGE 1ère année et de Master Informatique (spécialité ATAL) ont eu à modéliser en XML les données sur le projet suivant :

> *La société AtlanticMobileVideo (AMV) cherche à mettre en place un service de films à la demande. Elle met à disposition de ses clients un catalogue de films, d'acteurs et de réalisateurs. Les compositeurs de BO (Bande Originale) sont aussi présents. Nous ne nous occuperons pas, ici, des producteurs (sociétés ou individus).*
>
> *Les acteurs jouent dans des films. Certains films peuvent avoir plusieurs réalisateurs. Un acteur peut être réalisateur de films. Outre les acteurs, réalisateurs et compositeurs (appelés "artistes" dans la suite), les films sont décrits par leur titre (pas nécessairement original), leur durée, un résumé (en XHTML), le(s) pays de production, le genre (un seul par film) et leur classification (-12, -16, etc.). Pour chaque acteur qui joue dans un film, on connait aussi le nom de son (ou ses) rôle(s).*
>
> *Les clients parcourent le catalogue pour consulter la biographie (en XHTML) des artistes ou pour visionner un film particulier. Les clients (dont on connait l'identité et qui sont identifiés par un pseudo) peuvent souscrire à différentes offres selon le service attendu (par ordre de services croissant : "gratuit", "prémium standard" et "prémium universel"). Un client ne peut souscrire à deux offres en même temps.*
>
> *AMV désire aussi constituer un réseau social de cinéphiles. Aussi, les clients peuvent :*
>
> - *se lier à d'autres clients s'ils ont souscrit au l'offre "prémium standard" ou "prémium universel" ;*
> - *se déclarer "fan" d'un artiste ou apprécier un film en lui mettant une note (étoile) de 0 à 5, avec un commentaire textuel éventuellement, seulement s'ils ont souscrit l'offre "prémium universel".*
>
> *Afin d'aider les clients à trouver amis, artistes ou films, un système de mots-clés est mis en place. Ce système permettra à AMV de proposer à un client des clients, artistes ou films qui pourraient l'intéresser.*

AMV envisage aussi de proposer à ses clients des offres spéciales, par exemple leur anniversaire ou l'anniversaire de leur inscription. Les clients nés à la même date qu'un artiste (ou à la date de sortie d'un film) dont ils sont fans ont droit à une remise.

Annexe B : Schéma DTD de l'exemple

On obtient alors le schéma suivant :

```
<?xml version="1.0" encoding="UTF-8"?>

<!ENTITY % xhtml
                SYSTEM
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
%xhtml;

<!ELEMENT AMV (liste-types-client, liste-films, liste-artistes) >

<!-- Gestion des clients -->

<!ELEMENT liste-types-client (client)* >
<!ELEMENT client ((gratuit | prémium-standard | prémium-universel),
                                liste-mots-clés?)>
<!ATTLIST client
    pseudo ID #REQUIRED
    nom-cl CDATA #REQUIRED
    prénom-cl CDATA #REQUIRED
    adresse CDATA #REQUIRED
    date-naissance-cl CDATA #REQUIRED
    date-inscription CDATA #REQUIRED >

<!ELEMENT gratuit EMPTY >
<!ELEMENT prémium-standard (liste-amis) >
<!ELEMENT prémium-universel (liste-amis, liste-fans) >

<!ELEMENT liste-amis (ami*) >
<!ELEMENT ami EMPTY >
<!ATTLIST ami avec IDREF #REQUIRED >

<!ELEMENT liste-fans (fan*) >
<!ELEMENT fan EMPTY >
<!ATTLIST fan de IDREF #REQUIRED >

<!-- Gestion des films -->
```

```
<!ELEMENT liste-films (genre)+ >
<!ELEMENT genre (film+) >
<!ATTLIST genre désignation CDATA #REQUIRED >
<!ELEMENT film (résumé, liste-likes, produit+, liste-mots-clés?) >
<!ATTLIST film
    no-f ID #REQUIRED
    titre-f CDATA #REQUIRED
    classification (0|10|12|16|18) '0'
    date-sortie CDATA #REQUIRED
    durée CDATA #REQUIRED
    URL-flux CDATA #REQUIRED >

<!ELEMENT résumé (%block;) > <!--  %block; contenu de <boby> -->

<!ELEMENT liste-likes (like*) >
<!ELEMENT like (#PCDATA) >
<!ATTLIST like
    client IDREF #REQUIRED
    date-like CDATA #REQUIRED
    stars (0|1|2|3|4|5) '0' >

<!ELEMENT produit EMPTY >
<!ATTLIST produit pays CDATA #REQUIRED >

<!-- Gestion des artistes -->

<!ELEMENT liste-artistes (artiste)+ >
<!ELEMENT artiste
   (biographie, liste-mots-clés?, (joue|réalise|compose)+) >
<!ATTLIST artiste
    no-a ID #REQUIRED
    nom-a CDATA #REQUIRED
    prénom-a CDATA #IMPLIED
    nationalité CDATA #REQUIRED
    date-naissance-a CDATA #REQUIRED >

<!ELEMENT biographie (%block;) >

<!ELEMENT joue EMPTY >
<!ATTLIST joue
    dans IDREF #REQUIRED
    rôle CDATA #REQUIRED >
```

```
<!ELEMENT réalise EMPTY >
<!ATTLIST réalise le-film IDREF #REQUIRED >

<!ELEMENT compose EMPTY >
<!ATTLIST compose la-bo-de IDREF #REQUIRED >

<!-- Eléments communs -->

<!ELEMENT liste-mots-clés (mot-clé+) >
<!ELEMENT mot-clé (#PCDATA) >
```

Annexe C : Exemple de graphe associé

La figure 16 présente un exemple de graphe associé à notre exemple.

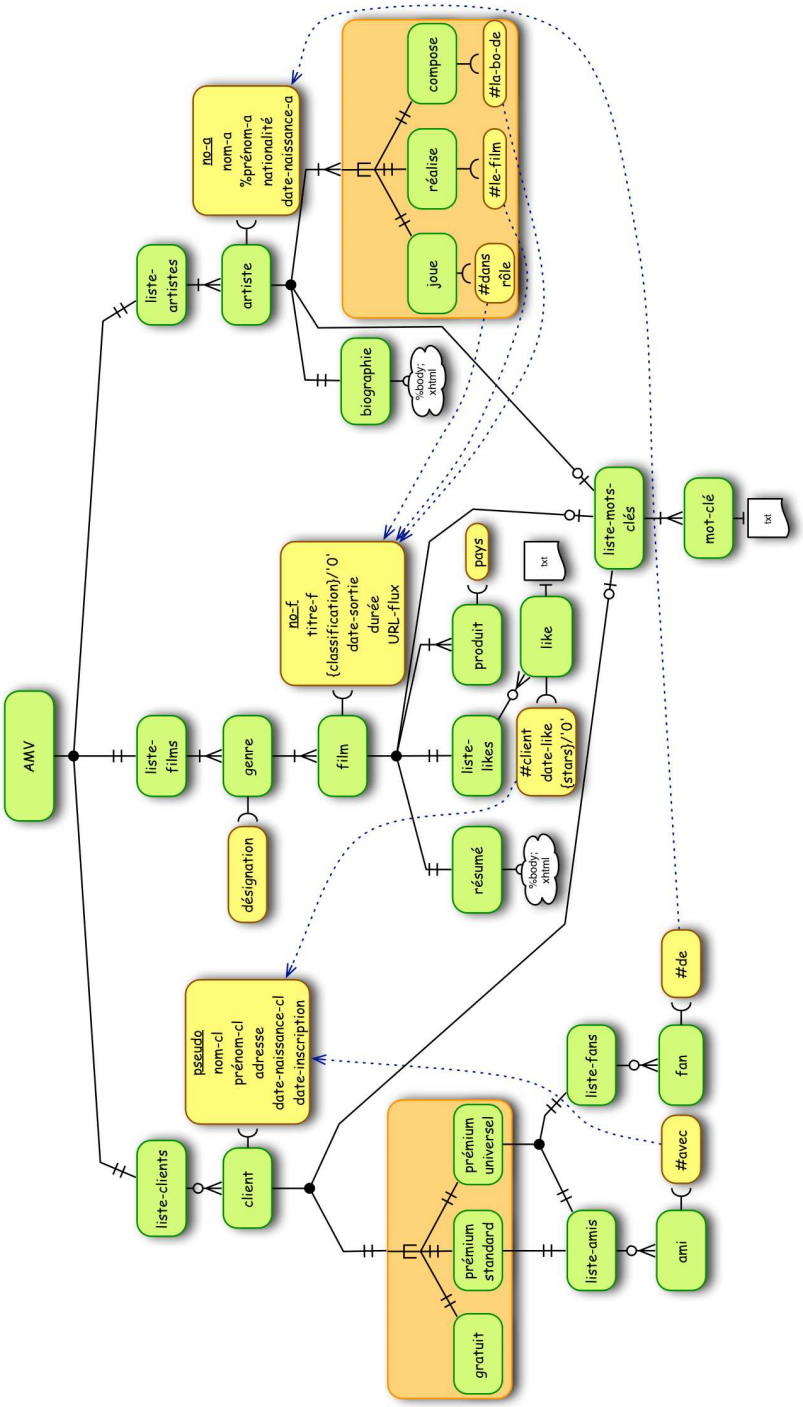


FIGURE 16: AMV

A Graph for the Learning of XML

Currently, XML is a format widely used. In the context of computer science teaching, it is necessary to introduce students to this format and, especially, at its eco-system. We have developed a model to support the teaching of XML. We propose to represent an XML schema as a graph highlighting the structural characteristics of the valide documents. We present in this report different graphic elements of the model and the improvements it brings to data modeling in XML.

Keywords: XML, Schema, DTD, XSD, Relax NG, Graph.